
PROTAGENTS: PROTEIN DISCOVERY VIA LARGE LANGUAGE MODEL MULTI-AGENT COLLABORATIONS COMBINING PHYSICS AND MACHINE LEARNING *

Alireza Ghafarollahi

Laboratory for Atomistic and Molecular Mechanics (LAMM)
Massachusetts Institute of Technology
77 Massachusetts Ave.
Cambridge, MA 02139, USA

Markus J. Buehler

Laboratory for Atomistic and Molecular Mechanics (LAMM)
Center for Computational Science and Engineering
Schwarzman College of Computing,
Massachusetts Institute of Technology
77 Massachusetts Ave.
Cambridge, MA 02139, USA

Correspondence: mbuehler@MIT.EDU

ABSTRACT

Designing *de novo* proteins beyond those found in nature holds significant promise for advancements in both scientific and engineering applications. Current methodologies for protein design often rely on AI-based models, such as surrogate models that address end-to-end problems by linking protein structure to material properties or vice versa. However, these models frequently focus on specific material objectives or structural properties, limiting their flexibility when incorporating out-of-domain knowledge into the design process or comprehensive data analysis is required. In this study, we introduce ProtAgents, a platform for *de novo* protein design based on Large Language Models (LLMs), where multiple AI agents with distinct capabilities collaboratively address complex tasks within a dynamic environment. The versatility in agent development allows for expertise in diverse domains, including knowledge retrieval, protein structure analysis, physics-based simulations, and results analysis. The dynamic collaboration between agents, empowered by LLMs, provides a versatile approach to tackling protein design and analysis problems, as demonstrated through diverse examples in this study. The problems of interest encompass designing new proteins, analyzing protein structures and obtaining new first-principles data – natural vibrational frequencies – via physics simulations. The concerted effort of the system allows for powerful automated and synergistic design of *de novo* proteins with targeted mechanical properties. The flexibility in designing the agents, on one hand, and their capacity in autonomous collaboration through the dynamic LLM-based multi-agent environment on the other hand, unleashes great potentials of LLMs in addressing multi-objective materials problems and opens up new avenues for autonomous materials discovery and design.

Keywords Multi-agent modeling · large language model (LLM) · physics-inspired machine learning · protein design

**Citation:* A. Ghafarollahi, M.J. Buehler, arXiv, DOI:000000/11111, 2024

1 Introduction

Proteins, the building blocks of life, serve as the fundamental elements of many biological materials emerging from natural evolution over the span of 300 million years. Protein-base biomaterials like silk, collagen and tissue assemblies such as skin exhibit diverse structural features and showcase unique combinations of material properties. The underlying sequences of amino acids (AAs) in a protein determines its unique three-dimensional structure, which, in turn, dictates its specific biological activity and associated outstanding properties. This inherent relationship has inspired scientists in the field of materials design and optimization to draw valuable insights from nature for creating novel protein-based materials. The diversity in protein design is immense, with over 20^{100} possible AA sequences for just a relatively small 100-residue protein. However, the natural evolutionary process has sampled only a fraction of this vast sequence space. This leaves a substantial portion uncharted, presenting a significant opportunity for the *de novo* design of proteins with potentially remarkable properties.[1] Despite this potential, the extensive design space, coupled with the costs associated with experimental testing, poses formidable challenges in *de novo* protein design. Navigating this intricate landscape necessitates the development of a diverse set of effective tools enabling the targeted design of *de novo* proteins with specific structural features or properties.

Over the past years, in the field of *de novo* protein design, data-driven and machine learning methods have emerged as powerful tools, offering valuable insights and accelerating the discovery of novel proteins with desired properties[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]. These methods have opened great avenues for predicting structure, properties, and functions of proteins solely based on their underlying AA sequence. For instance, the development of deep learning (DL)-based AlphaFold 2 marked a significant breakthrough in the field of 3D folding protein prediction with a level of accuracy that in some cases rivaled expensive and time-consuming experimental techniques.[16] Moreover, deep learning-based models have been developed to explore structure-property relationships in the analysis and design of proteins. These models encompass a broad spectrum of structural and mechanical properties, serving either as constraints or target values. For example, various DL-models developed predict the secondary structure of proteins from their primary sequences. Prediction of mechanical properties of spider silk protein sequences have been enabled by DL models[17, 18, 19, 20, 21, 22]. Moreover, DL-based models such as graph neural networks[23] and transformer-based language models[24] show enhanced accuracy in predicting the protein natural frequencies compared to physics-based all-atom molecular simulations. The development of such DL models significantly reduces the cost of screening the vast sequence space to target proteins with improved or optimized mechanical performance. In the field of *de novo* protein design, data-driven and machine learning methods have emerged as powerful tools, offering valuable insights and accelerating the discovery of novel proteins with desired properties[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]. These methods have opened great avenues for predicting structure, properties, and functions of proteins solely based on their underlying AA sequence. For instance, the development of deep learning (DL)-based AlphaFold 2 marked a significant breakthrough in the field of 3D folding protein prediction with a level of accuracy that in some cases rivaled expensive and time-consuming experimental techniques.[16] Moreover, deep learning-based models have been developed to explore structure-property relationships in the analysis and design of proteins. These models encompass a broad spectrum of structural and mechanical properties, serving either as constraints or target values. For example, various DL-models developed predict the secondary structure of proteins from their primary sequences. Prediction of mechanical properties of spider silk protein sequences have been enabled by DL models[17, 18, 19, 20, 21, 22]. Moreover, DL-based models such as graph neural networks[23] and transformer-based language models[24] show enhanced accuracy in predicting the protein natural frequencies compared to physics-based all atomistic simulations. The development of such DL models significantly reduces the cost of screening the vast sequence space to target proteins with improved or optimized mechanical performance.

A frontier, however, that still exists is how we can create intelligent tools that can solve complex tasks and draw upon a diverse set of knowledge, tools and abilities. Another critical issue is that the combination of purely data-driven tools with physics-based modeling is important for accurate predictions. Moreover, such tools should ideally also be able to retrieve knowledge from, for instance, the literature or the internet. All these aspects must be combined in a nonlinear manner where multiple dependent steps in the iteration towards and answer are necessary to ultimately provide the solution to a task. As we will discuss in this study, such an integration of tools, methods, logic, reasoning and iterative solution can be implemented through the deployment of a multi-agent system driven by sophisticated Large Language Models (LLMs).

LLMs[25, 26] have represented a paradigm shift in modeling problems across a spectrum of scientific and engineering domains[27, 28, 29, 30, 8, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41]. Such models, built upon attention mechanism and transformer architectures[42], have emerged as powerful tools recently in the field of materials science and related areas, contributing to various aspects ranging from knowledge retrieval to modeling, design, and analysis. For example, models such as ChatGPT and the underlying GPT-4 architecture[43], part of the Generative Pretrained Transformer (GPT) class, demonstrate exceptional proficiency in mastering human language, coding[44], logic and reasoning[45]. Recent

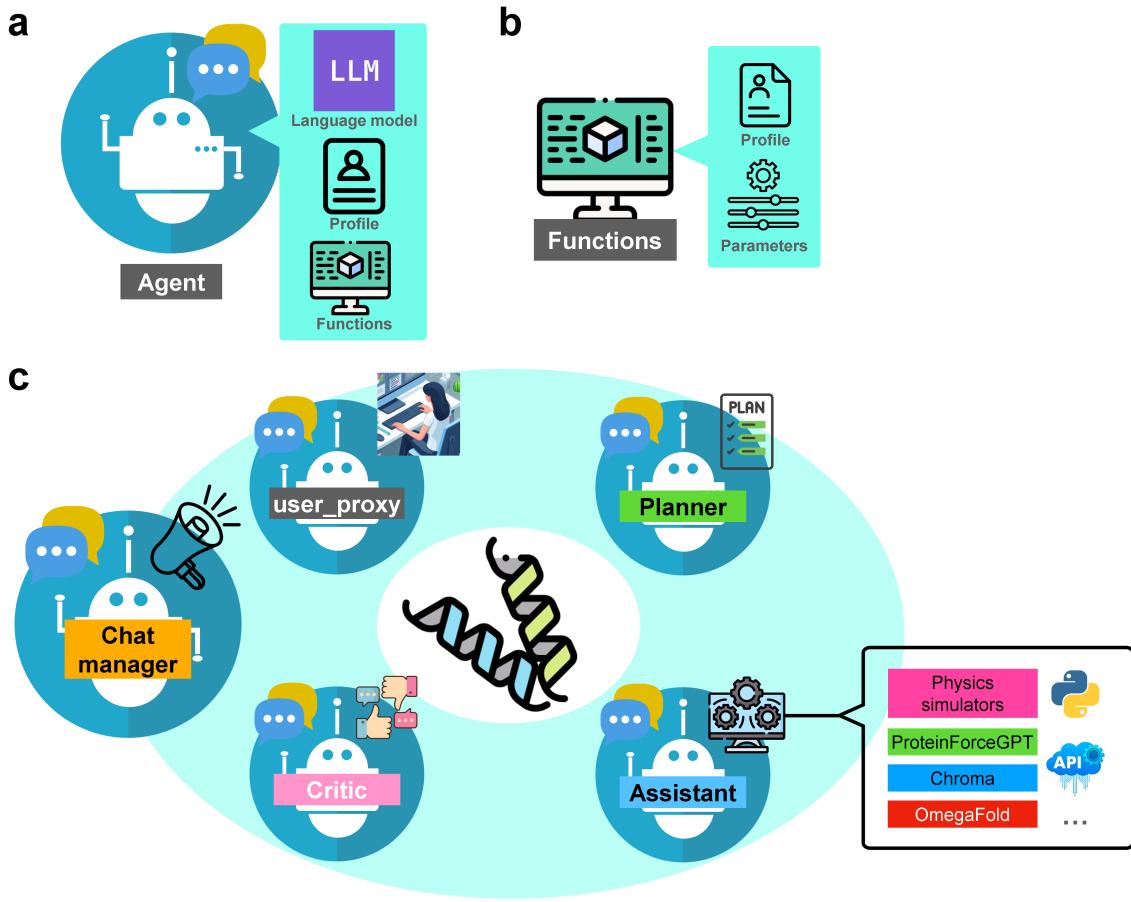


Figure 1: Multi-agent AI framework for automating protein discovery and analysis. **a**, A genetic agent structure in a multi-agent modeling environment that can communicate via language, has a focus defined by a profile, and has access to custom functions. **b**, A function is customized by a profile and a set of parameters. **c**, The structure of a team of agents, each with special expertise, that communicate to each other and allow for mutual correction and a division of labor. Given different profiles for each agent, agents are designed that are expert on describing the problem (*user_proxy*), plan making (*planner*), function executing (*assistant*), and result evaluation (*critic*). The whole process is automated via a dynamic group chat under the leading chat manager, offering a versatile approach in solving challenging tasks in the context of protein design and analysis without human intervention.

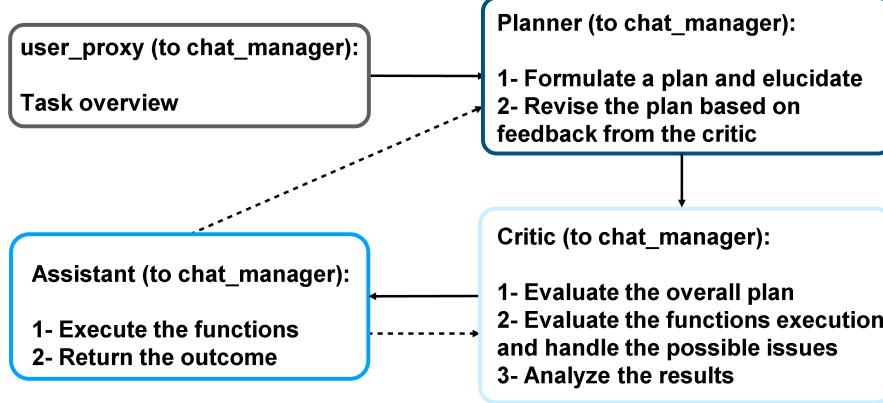


Figure 2: A generic flowchart showing the dynamic interaction between the multi-agent team members organized by the group chat manager to solve protein design and analysis problems. The manager selects the working agents to collaborate in the team work based on the current context of the chat, thus forming close interactions and enabling mutual corrections.

studies highlight their ability to proficiently program numerical algorithms and troubleshoot code errors across several programming languages like Python, MATLAB, Julia, C, and C++[46]. The GPT class of LLMs has also represented a new paradigm in simulating and predicting the materials behavior under different conditions[28], a field of materials science often reserved for conventional deep learning frameworks[47] such as Convolutional Neural Networks[48, 49], Generative Adversarial Networks[50, 51, 52], Recurrent Neural Networks[22, 54, 55][20, 53, 54], and Graph Neural Networks[23, 55, 56, 57, 58]. Moreover, due to their proficiency in processing and comprehending vast amount of different types of multimodal data, LLMs show promising capabilities in materials analysis and prediction application including key knowledge retrieval[35], general language tasks, hypothesis generation[29], and structure-to-property mapping[28, 59].

At the same time, LLMs are typically not best equipped to solve specific physics-based forward and inverse design tasks, and are often focused on leveraging their conversational capabilities. Here, LLMs have been instrumental in powering conversable AI agents, facilitating the transition from AI-human conversations to AI-AI or AI-tools interactions for increased autonomy.[31, 35, 60, 61, 62] This capability represents a significant advancement, enabling intelligent mediation, fostering interdisciplinary collaboration, and driving innovation across disparate domains, including materials analysis, design, and manufacturing. The overall process could be deemed as adapting a problem-solving strategy dictated and directed by the AI system comprised of different agents. Thereby, the entire process can be AI automated with reduced or little human intervention. Depending on the complexity of the problem, using the idea of labor division, the agents have the capability to break the overall task into subtasks for which different agents or tools are used consecutively to iteratively solve the problem until all subtasks have accomplished and the solution has achieved. There is no intrinsic limitation in defining the type of tools, making the multi-agent model a versatile approach in addressing problems across scales and disciplines. The tools could range from a simple linear mathematical function to sophisticated deep neural network architectures. The multi-agent strategy has been explored in materials and mechanics applications through in earlier work[29] and was further explored in the context of molecular modeling tasks[35].

In this paper, we propose a multi-agent strategy to the protein design problems by introducing ProtAgents, a multi-agent modeling framework to solve protein-related analysis and design problems by leveraging customized functions across domains and disciplines. The core underpinning concept of the multi-agent systems is the use state-of-the-art LLMs combined with a series of other tools. The LLM backbone demonstrate exceptional abilities in analysis, rational thinking, and strategic planning, essential for complex problem-solving. Leveraged by these capabilities, the proposed model aims to reduce the need for human intervention and intelligence at different stages of protein design. The agent model consists a suite of AI and physics based components such as:

- Physics simulators: obtain new physical data from simulations, specifically normal modes and vibrational properties by solving partial differential equations (PDEs)
- Generative AI model: conditional/unconditional *de novo* protein design, based on a denoising diffusion model

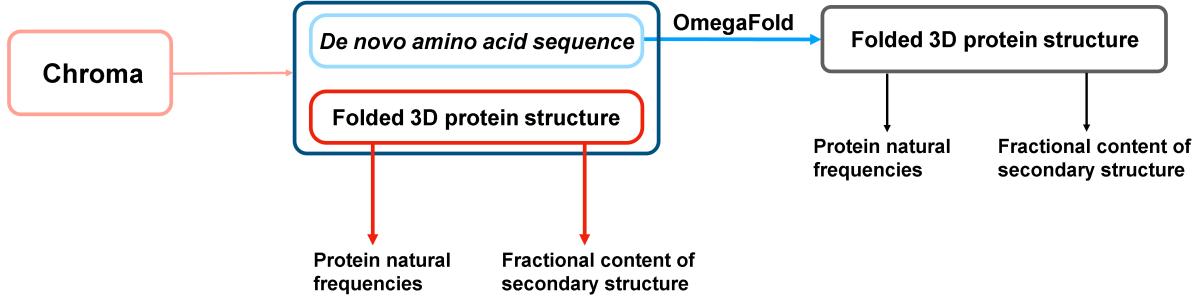


Figure 3: Overview of the multi-agent work to solve the complex task posed in experiment II, Section 2.2. First the multi-agent uses Chroma to generate *de novo* protein sequences and then computes natural frequencies and secondary structures content for the generated structures. Next, from *de novo* AA sequences, the model finds the 3D folded structures using OmegaFold and finally computes the frequencies and secondary structure content for the protein structures. The results obtained from the Chroma and OmegaFold 3D protein structures are compared in Figure 5.

- Fine-tuned transformer model: predict mechanical properties of proteins from their sequence
- Retrieval agent: retrieve new data from a knowledge database of scientific literature

The resulting model has the ability to go beyond the conventional DL models by integrating new physical data or information across disciplines, for instance via writing and executing code that solves differential equations or other physics-based numerical methods, or that conducts retrieval augmented generation (RAG)[63]. A tool-baked agent has access to various tools and functions with different functionalities that may be called upon, for instance, to predict a specific protein property or to obtain new physical data such as natural frequency from physics-based simulations. The versatility of the approach in solving complex tasks is exhibited by providing a series of experiments in the context of proteins design, modeling, and data analysis.

The plan of this paper is as follows. In Section 2, we present an overview of the multi-agent framework developed to tackle multi-objective complex tasks. Subsequently, we delve into a series of experiments where each task is initially introduced, followed by a detailed examination of various aspects throughout the problem-solving process by the multi-agent teamwork. A comprehensive discussion regarding the multi-agent framework and future prospects is provided in Section 3.

2 Results and Discussion

We present a series of computational experiments aimed at evaluating the effectiveness and potential of a multi-agent modeling framework for various challenges within the domain of protein modeling, design, and analysis. The multi-agent framework consists of a team of agents, each powered by a state-of-the-art general purpose large language model, GPT-4,[43] accessed via the OpenAI API[64] and characterized by a unique profile that details its role, and communication protocols, such as sharing information and engaging with humans via language as shown in Figure 1a. Furthermore, agents are given access to a set of tools with various functionalities across domains. As shown in Figure 1b each function is characterized by a descriptive profile and input parameters. The outline of the proposed multi-agent framework is shown in Figure 1c, illustrating the collaborative efforts of a team of agents with the following entities

- “User”: human that poses the question
- “Planner”: develops a plan to solve the task. Also suggests the functions to be executed.
- “Assistant”: who has access to all the customized functions, methods, and APIs and executes them to find or compute the relevant data necessary to solve the task
- “Critic”: Responsible for providing feedback about the plan developed by “planner” as well as analyzing the results and handling the possible mistakes and providing the output to the user.

The agents are organized into a team structure, overseen by a manager who coordinates overall communication among the agents. A generic structure showing the dynamic collaboration between the team of agents proposed in the current study is depicted in Figure 2. Moreover, Table 1 lists the full profile for the agents recruited in our multi-agent framework. Further details can be found in the **Materials and Methods** section 4.

Table 1: The profiles of the agents implemented in the current study to solve multi-objective tasks in the context of protein design and analysis.

Agent #	Agent role	Agent profile
1	user_proxy	user_proxy. Plan execution needs to be approved by user_proxy
2	Planner	Planner. You develop a plan. Begin by explaining the plan. Revise the plan based on feedback from the critic and user_proxy, until user_proxy approval. The plan may involve calling custom function for retrieving knowledge, designing proteins, and computing and analyzing protein properties. You include the function names in the plan and the necessary parameters. If the plan involves retrieving knowledge, retain all the key points of the query asked by the user for the input message.
3	Assistant	Assistant. You have access to all the custom functions. You focus on executing the functions suggested by the planner or the critic. You also have the ability to prepare the required input parameters for the functions.
4	Critic	user_proxy. You double-check the plan, especially the functions and function parameters. Check whether the plan included all the necessary parameters for the suggested function. You provide feedback.
5	Group chat manager	You repeat the following steps: Dynamically selecting a speaker, collecting responses, and broadcasting the message to the group.

It is noteworthy that critical issues in the realm of protein design surpass the capabilities of mere Python code writing and execution. Instead, addressing these challenges necessitates the utilization of external tools specifically tailored for protein design and analysis, and the writing, adaptation, correction and execution of code depends nonlinearly on the progression of the solution strategy that is developed by the system.

The tools are incorporated into the model via the Assistant agent who oversees executing the tools. To assess the performance of the multi-agent framework in handling complex interdisciplinary tasks, we have defined a rich library of functions each with special powers in solving the protein problems. Each function has a distinct profile describing its functionality and takes one or more required entities as the input. The functions provide the ability to, for instance, retrieve knowledge, perform protein folding, analyze the secondary structure, and predict some parameters through a pre-trained autoregressive language model. Additionally, a function can carry out simulations to compute the protein natural frequencies, thus allowing the model to integrate the new physics-based data. A full list of functions implemented in the current study is provided in **Table S1** in the supplementary material.

Given the complexities residing in the protein design problems, the primary contribution of our multi-agent framework lies in assessing whether the team of agents can discern the requisite tools for a given query and evaluating the framework’s capability to initiate the execution of these tools, along with providing the necessary inputs. The designated tasks are intentionally designed to be sufficiently complex, involving multiple subtasks where, for some cases, the execution of each depends on the successful completion of the preceding ones. This design showcases the model’s capacity for the automated handling of intricate tasks, eliminating or substantially reducing the need for human intervention. Although the multi-agent framework allows for the human intervention at different stages, we skip that to further examine the team’s capability in handling different possible situations, for instance in case of a failure.

2.1 Experiment I: Knowledge retrieval, computations, and analysis

As the first example, we pose a task which involves a multi-step knowledge retrieval, analysis, and computations for a set of protein PDB identifiers. We start the first round of conversation by posing the following question:

user_proxy (to chat_manager):

Can you provide examples of protein names whose mechanical properties have been studied through experiments?

The planer then correctly suggests the function “retrieve_content” to be executed with the argument “examples of protein names whose mechanical properties have been studied through experiments”. Upon execution of the function, the assistant provides us with a list of protein names. Upon inspection, we find that the agent has successfully identified experimentally studied proteins, despite an abundance of information on proteins studied theoretically, for instance, through coarse-grain simulations. Since we are interested in the PDB ids, we continue the chat by a follow-up question “Can you provide me with the PDB ids for these proteins?” when “user_proxy” is being asked to provide feedback to chat manager. Again, the planner suggests “retrieve_content” function with the following message

Planner (to chat_manager):

```
***** Suggested function Call: retrieve_content ***** Arguments: { "message": "PDB ids for I27 domain from titin, fibronectin type III domains, E2lip3, T4 lysozyme, ubiquitin, titin immunoglobulin domains, cell adhesion molecule, spectrin repeats, and tenascin" }
```

The “Assistant” agent then calls the function and gives the following output:

Assistant (to chat_manager):

Here are the PDB IDs for the proteins you mentioned:

1. I27 domain from titin: 1wit
2. Ubiquitin: 1ubq
3. T4 lysozyme: 1nct
4. Titin immunoglobulin domains: 1tit, 1qjo
5. E2lip3: 1hz6/2ptl
6. Tenascin: 1ten
7. Spectrin repeats: 1emb
8. FnIII: 1g1c, 1fnh, 1vsc, 1cfc, 1fnf/1ttf/1ttg
9. Cell adhesion molecule: 1bni/1bnr, 1b6i, 1rsy/1dqv, 1aj3, 1ksr/1whl, 1u4q, 1hci, 1n11

Upon careful examination of the results, we observe that, despite all the PDB ids exist in the source database, the PDB ids do not quite match with the protein names except for a few cases. Nevertheless, note that the error is caused by the poor performance of the “*retrieve_content*” function and the team of agents cannot mitigate that as they have no access to the knowledge database. As such, we continue to test the agent-team capability in more challenging queries centered around computational tasks and physics-based simulations by assigning the following task in the next round of conversation

user_proxy (to chat_manager):

For the first 10 protein PDB ids in the list, if the AA sequence length of the protein is less than 128, do the following: (a) analyze the secondary structure of the protein, (b) compute its first 13 natural frequencies, (c) determine its structure classification.

The above is a complex multi-step analysis and computation task encompasses aspects such as secondary structures analysis, natural frequency calculations, and structure classification. Additionally, the task is subject to an initial condition that must be satisfied before proceeding through the next sequence of steps, adding an extra layer of complexity. In response, the planner comes up with a detailed plan which consists of all the actions that need to be taken to complete the task. Moreover, the plan mentions all the corresponding functions that need to be executed to accomplish the task. More importantly, the “*planner*” perfectly realizes to fetch the protein structures first before starting to analyze the secondary structure, although it was not explicitly mentioned in the task query.

The teamwork proceeds by a follow-up feedback provided by the “*critic*” agent about all the plan steps and functions which is concluded by the following statement

critic (to chat_manager):

Overall, the plan rightly identifies all the necessary functions and respective parameters for each step. It illustrates a good understanding of what needs to be done to fulfill the user request. Good job!

Therefore, the positive feedback from the “*critic*” further supports the good performance of the planner in addressing all the critical steps required to accomplish the tasks.

The “*assistant*” agent then follows the plan by calling and executing the corresponding functions, starting with AA length calculation, until all the steps have been undertaken. The results show that all the inputs to the functions are properly identified and provided and the functions are executed without any error. The conditional statement included in the tasks is also correctly satisfied for each protein, that is the computations are conducted only if the sequence length is less than 128 and are emitted otherwise. For instance, for the protein with PDB id “1hz6” the AA length is returned as 216 by the “*assistant*” which is then followed by the following message from the “*critic*”

critic (to chat_manager):

The length of the amino-acid sequence for the PDB id ‘1hz6’ is 216, which is greater than 128. Therefore, we will not perform the steps (a) analyze the secondary structure of the protein, (b) compute its first 13 natural frequencies, and (c) determine its structure classification for the ‘1hz6’.

Moving to the next PDB id in the list...

Table 2: The results generated by the group chat in the form of a CSV file, without human intervention, for the experiment I, Section 2.1.

Protein ID #	Amino Acid Length	Secondary Structure	First 13 Frequencies	CATH Classification
Iwit	93	[{'H': 0.0, 'B': 3.23, 'E': 51.61, 'G': 3.23, 'I': 0.0, 'T': 13.98, 'S': 5.38, 'P': 0.0, '-': 22.58}]	[4.3755, 5.0866, 5.5052, 6.7967, 7.908, 8.1947, 9.0166, 9.8528, 11.0632, 11.3968, 11.7355, 12.1279, 12.3498]	2.60.40.10
Iubq	76	[{'H': 15.79, 'B': 2.63, 'E': 31.58, 'G': 7.89, 'I': 0.0, 'T': 15.79, 'S': 5.26, 'P': 5.26, '-': 15.79}]	[0.7722, 1.0376, 1.5225, 1.6534, 2.5441, 2.9513, 3.2873, 3.7214, 4.1792, 4.3437, 4.3908, 4.6551, 5.1631]	3.10.20.90
Inct	106	[{'H': 0.0, 'B': 4.08, 'E': 35.71, 'G': 0.0, 'I': 0.0, 'T': 2.04, 'S': 21.43, 'P': 0.0, '-': 36.73}]	[3.6644, 4.425, 6.5351, 6.7432, 7.1409, 7.1986, 9.0207, 9.2223, 10.3163, 10.7313, 11.5299, 11.6373, 12.5606]	2.60.40.10
Itit	98	[{'H': 0.0, 'B': 1.12, 'E': 35.96, 'G': 0.0, 'I': 0.0, 'T': 6.74, 'S': 17.98, 'P': 0.0, '-': 38.2}]	[5.5288, 5.9092, 8.2775, 8.6267, 9.3391, 9.8783, 10.1607, 11.451, 11.5896, 11.7052, 12.1498, 12.6082, 13.8622]	2.60.40.10
1qjo	80	[{'H': 0.0, 'B': 2.5, 'E': 40.0, 'G': 0.0, 'I': 0.0, 'T': 8.75, 'S': 13.75, 'P': 0.0, '-': 35.0}]	[3.8578, 4.4398, 5.4886, 5.7815, 6.6332, 6.9269, 7.2329, 7.6453, 8.2545, 8.3076, 8.6118, 8.7135, 8.8546]	2.40.50.100
2ptl	78	[{'H': 15.38, 'B': 1.28, 'E': 30.77, 'G': 0.0, 'I': 0.0, 'T': 7.69, 'S': 19.23, 'P': 0.0, '-': 25.64}]	[0.0386, 0.1161, 0.2502, 0.5921, 1.1515, 1.5257, 2.0924, 2.6793, 3.4292, 3.9289, 4.2172, 4.6878, 4.8022]	3.10.20.10

After completion of all the tasks, the assistant returns a summary of all the results for each protein as representatively shown below for PDB id “*Iwit*”:

assistant (to chat_manager):

For PDB id ‘Iwit’:

- It has an amino acid sequence length of 93.
- The analysis of its secondary structure yielded the following composition: H: 0%, B: 3.23%, E: 51.61%, G: 3.23%, I: 0%, T: 13.98%, S: 5.38%, P: 0%, -: 22.58%.
- Its first 13 natural frequencies are [4.3755, 5.0866, 5.5052, 6.7967, 7.908, 8.1947, 9.0166, 9.8528, 11.0632, 11.3968, 11.7355, 12.1279, 12.3498].
- Its CATH classification is 2.60.40.10.

The results suggest that the framework effectively retains all outputs, demonstrating its strong memory even in the face of diverse and extended results. As the last round of conversation, we ask to save all the results which allows us to load them at later time for other purposes:

user_proxy (to chat_manager):

Save all the results into a csv file where each row corresponds to a protein and each column represent a protein property, utilizing proper identifiers.

In response, the planner suggests to call the python function “*save_to_csv_file*”. The main task here is to generate the dictionary of results in JSON and with appropriate structure as instructed by the user. However, we see that upon generating the JSON data and inputting it into the function by the “*assistant*” agent, the following error occurs:

```
***** Response from calling function "save_to_csv_file" *****
Error: Expecting ',' delimiter: line 1 column 141 (char 140)
```

Without any human intervention, the agent team is able to resolve the issue by mutual correction. In particular, the “*critic*” identifies the cause of error by writing

Critic (to chat_manager):

It seems like there is an error in the JSON format being used to feed data into the csv conversion function. The JSON string appears to be improperly formed leading the function to fail. The values for “Secondary Structure” and “First 13 Frequencies” are stringified JSON objects themselves, which maybe causing

The “*assistant*” then reconstructs the JSON file from the output results and is able to successfully execute the function and thus save the results in a csv file as shown in **Table 2**. The complete group chat records can be found in **Table S2** of the Supplementary material.

2.2 Experiment II: *De novo* protein design using Chroma

An important characteristic of the multi-agent model is its capability in handling very complex tasks in the context of protein design and analysis. This partly stems from the possibility of incorporating customized functions in the model for various purposes from knowledge retrieval to performing physics-based simulations. In this example, we explore the model's performance in using the state-of-the art models in *de novo* protein design, Chroma[15], and protein folding, OmegaFold[4]. Chroma offers a unified approach for protein design that can directly sample novel protein structures and sequences with the possibility to direct the generative design towards the desired functions and properties by appropriate conditioning. OmegaFold is a state-of-the-art folding method without the need of using multiple sequence alignments (MSA) to predict the three-dimensional folded protein structure given its AA sequence.

In this experiment, we formulate a complex multi-step task with the objective of comparing the two models based on various structural and physical features derived from the folded structures obtained through Chroma and OmegaFold2. We pose the following task through the “*user_proxy*” agent:

user_proxy (to chat_manager):

Design three proteins with sequence length of 120 and analyze their secondary structure and first 10 frequencies. Then fold the proteins and repeat the analysis. Save all the results (including the AA sequences) into a csv file where each row corresponds to a protein and each column represents a protein property, utilizing proper identifiers.

The “*planner*” then suggests the following plan

Planner (to chat_manager):

Plan: 1. Use the ‘design_protein_from_length’ function three times to design three proteins each with a sequence length of 120. I will use 1, 2, 3 as names for the proteins. Save the amino acid sequences and PDB filenames.

2. Use the ‘analyze_protein_structure’ function to analyze the secondary structure of the proteins. Store the results.

3. Use the ‘calc_protein_ANM’ function to calculate the first 10 frequencies of the proteins. Store the results. 4. Use the ‘fold_protein’ function to fold the proteins.

5. Repeat steps 2 and 3 with the folded proteins.

6. Use the ‘save_to_csv_file’ function to save all results into a csv file where each row corresponds to a protein and each column represents a protein property.

7. The identifiers will be named as follows:

- Protein Name*
- Amino Acid Sequence (Pre-Fold)*
- Secondary Structure (Pre-Fold)*
- Frequencies (Pre-Fold)*
- Amino Acid Sequence (Post-Fold)*
- Secondary Structure (Post-Fold)*
- Frequencies (Post-Fold)*

At the first glance, the plan seems to cover all the details necessary to accomplish the tasks included in the problem statement. However, the “*critic*” agent who is responsible in giving feedback about the plan spots a minuscule error in the saving part of the plan as follows:

Critic (to chat_manager):

Your plan is almost perfect. However, when a protein is folded, the protein sequence doesn't change, only the structure does, which then affects the secondary structure and possible frequencies you can calculate. So you need to only save Amino Acid Sequence once after designing the proteins. Each protein should have the following column properties:

- Protein Number*
- Amino Acid Sequence*
- Secondary Structure (Pre-Fold)*
- Frequencies (Pre-Fold)*

- Secondary Structure (Post-Fold)
- Frequencies (Post-Fold)

Other than that, your plan works perfectly. So steps will be as follows:

...

The correction made by the “*critic*” concerning the sequence length underscores its notable proficiency in comprehending how diverse functions and parameters influence various aspects within the realm of protein design.

The “*user_proxy*” agent is then asked to confirm the plan. The “*assistant*” then takes the stage and starts following the plan by calling and executing the functions until all the steps have been undertaken. An overview of the work performed by the “*assistant*” is depicted in Figure 3. At the end of the computations, the results are formatted into a JSON dictionary to feed into the “*save_to_csv_file*” function. However, an error related to the JSON dictionary format occurs when executing the function as follows:

```
***** Response from calling function "save_to_csv_file" *****
Error: Expecting property name enclosed in double quotes: line 2 column 5 (char 6)
*****
```

The “*critic*” then steps in by making the following comment and suggesting a plan to fix the error as follows:

Critic (to chat_manager):

I apologize for the mistake. It seems there was an issue with the formatting of the input JSON dictionary for saving to the CSV file. Let's try again with properly formatted JSON.

Plan to fix the error:

1. *Correctly format JSON with double quotes instead of single quotations.*
2. *Call 'save_to_csv_file' function again with correctly formatted JSON dictionary.*

I'll directly proceed with this plan as the error was due to syntax not plan.

The critic makes the necessary corrections and suggest the corrected JSON dictionary for the “assistant” to execute the “*save_to_csv_file*” function. This time, the function is successfully executed and the results are saved into a csv file as shown in **Table 3**. At last, the “*critic*” gives an evolution about the whole process:

Critic (to chat_manager):

The plan was executed successfully and all data including Amino Acid Sequence, Secondary Structure (Pre-Fold and Post-Fold), and Frequencies (Pre-Fold and Post-Fold) for three proteins have been saved into a CSV file named 'protein_data.csv'. Each row of the file corresponds to a protein and each column represents a protein property.

This experiment showcases a good example of multi-agent collaboration in solving complex tasks in the context of *de novo* protein design and analysis. It specially shows the great capability of the “*critic*” agent in providing valuable feedback to other working agents at different stages of the problem solving endeavor, further assisting the team of agents in handling possible errors without the need for human involvement. Figure 5 shows the plots of the generated results including the 3D folded structures. The full conversations can be found in **Table S3** of the Supplementary material.

2.3 Experiment III: Protein design conditioned on the protein CATH class

CATH is a hierarchical classification system for protein structures that consists of four main levels. The highest level in this hierarchy is the “*Class*” which primarily characterizes the secondary structure content of the protein. For example, C_1 , C_2 , and C_3 correspond to proteins predominantly composed of α -helix, mainly β -sheet, and a combination of α and β secondary structures. Consequently, designing proteins based on the CATH class number, i.e. C_1 , C_2 , C_3 , can be understood as creating proteins with a specific fractional content of the secondary structure. Previous studies have demonstrated the importance of the protein secondary structures content, specially α -helix/ β -sheet ratio, on the mechanical properties of the protein materials[65, 66]. For instance, α -helix-rich proteins tend to yield stretchy materials[67], while β -sheet-rich ones produce rigid materials.[68, 69, 70] Chroma has the potential to conditionally generate proteins with specified folds according to CATH class annotations at three levels.[15]

Table 3: The final results generated by the group chat in the form of a CSV file, without human intervention, for the second experiment II, Section 2.2.

Protein Number #	Amino Acid Sequence	Secondary Structure (Pre-Fold)	Frequencies (Pre-Fold)	Secondary Structure (Post-Fold)	Frequencies (Post-Fold)
1	MIINIKTENGLSITYNSID EKKLELKTYTPVKSPEDFK FPEDAKATISEVEYKGKK VIKIDAKLYVSPDLSKAK LTIEVNADISQEEADKIID EFLKLLESLGNIKLKVTK DGNKYTIIEV	'H': 13.3333333333, 'B': 0.0, 'E': 46.6666666666, 'G': 0.0, 'I': 0.0, 'T': 14.1666666666, 'S': 7.5, 'P': 0.0, '-' 18.3333333333	[2.0337, 2.8678, 3.3843, 3.6263, 3.9904, 4.5381, 4.8373, 4.8956, 5.1492, 5.4416]	'H': 15.8333333333, 'B': 0.0, 'E': 46.6666666666, 'G': 2.5, 'I': 0.0, 'T': 14.1666666666, 'S': 4.1666666666, 'P': 0.0, '-' 16.6666666666	[1.8739, 2.1563, 2.7611, 3.1086, 3.8712, 4.0481, 4.3759, 4.6717, 4.8183, 4.9126]
2	GSPLPPRPLSPEEQEARL KKAQEKYNEFVSKIKEL LRRAADRVRGEPEVLEIE TKIGDYEYKIVATSPEE AKELENLIKEMIDLGFKP SKEFSDKLVEAARLIREG RVDEALRLDEM	'H': 61.6666666666, 'B': 0.0, 'E': 11.6666666666, 'G': 0.0, 'I': 0.0, 'T': 7.5, 'S': 3.3333333333, 'P': 3.3333333333, '-' 12.5	[0.0207, 0.1058, 0.1782, 0.4189, 0.49, 0.9015, 1.1832, 1.8257, 2.1212, 2.8726]	'H': 62.5, 'B': 0.0, 'E': 11.6666666666, 'G': 0.0, 'I': 0.0, 'T': 6.6666666666, 'S': 1.6666666666, 'P': 4.1666666666, '-' 13.3333333333	[0.0444, 0.1641, 0.3379, 0.5724, 0.765, 0.9568, 1.4306, 1.5344, 1.6834, 1.8099]
3	APLDPPDLSAQRLRAIDE LVRLLGYEEEVSKPEFIEA LRLYALDLGLKEVVLRR VTPAPASQPVGVTVEDV TVDEALRKQELSPPEQA RLEKIRAKYDEMLADPE FQALLDEVALARAAA	'H': 57.4999999999, 'B': 0.0, 'E': 13.3333333333, 'G': 0.0, 'I': 4.1666666666, 'T': 8.3333333333, 'S': 3.3333333333, 'P': 6.6666666666, '-' 6.6666666666	[0.7546, 1.0836, 1.5026, 1.8874, 2.0844, 2.3192, 2.7975, 3.0199, 3.0669, 3.1382]	'H': 61.6666666666, 'B': 0.0, 'E': 15.0, 'G': 0.0, 'I': 0.0, 'T': 8.3333333333, 'S': 3.3333333333, 'P': 1.6666666666, '-' 10.0	[0.5256, 1.0278, 1.1566, 1.2877, 1.5521, 1.9111, 2.1887, 2.4664, 2.734, 2.8731]

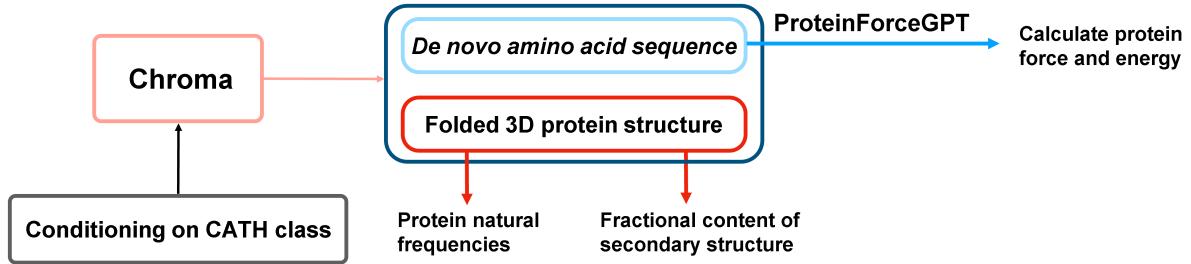


Figure 4: Overview of the multi-agent work to solve the complex task posed in experiment III, Section 2.3. First the multi-agent uses Chroma to generate *de novo* protein sequences and structures conditioned on the input CATH class. Then using the generated protein structures, the natural frequencies and secondary structures content are computed. Next, the force (maximum force along the unfolding force-extension curve) and energy (the area under the force-extension curve) are computed from *de novo* AA sequences using ProteinForceGPT.

In this example, we task the multi-agent team with generating proteins based on their fractional content of the secondary structure and subsequently performing computational and structural analysis tasks. Specifically, in addition to secondary structure analysis and natural frequency calculations, as covered in previous examples, we instruct the team to compute the maximum unfolding force (maximum force in the unfolding force-separation curve) and unfolding energy (the area under the unfolding force-separation curve) for each generated protein. To accomplish the latter, we have equipped the multi-agent team with a custom function that utilizes a trained autoregressive transformer generative AI model, *ForceGPT*. In addition to maximum unfolding force and energy, the trained generative model is able to predict the entire unfolding force-separation curve based solely on the protein amino acid sequence. Furthermore, the model has the capability to perform inverse design tasks by generating protein AA sequences that yield desired unfolding behavior. Detailed information about the training of the model can be found in **Materials and Methods** section4. The task given is:

user_proxy (to chat_manager):

Design two proteins of length 100 for each family of mainly beta, mainly alpha, and alpha beta proteins. Then compute the maximum force and unfolding energy, analyze the secondary structure and compute the first 10 frequencies for these proteins.

Note that, as before, we do not specify any particular function or offer hints for selecting the appropriate function to accomplish the tasks. Instead, we empower the agents to formulate a plan, wherein they decide which functions to select and determine the input parameters. The planner outlines the following plan for the given task:

Planner (to chat_manager):

1. Use the "design_protein_from_CATH" function with inputs of length of 100, CATH_ANNOTATION of 1 (mainly alpha), and a protein name for the first mainly alpha protein. Repeat with a different name for the second protein.
2. Use the same function again with inputs of length 100, CATH_ANNOTATION of 2 (mainly beta), and a protein name for the first mainly beta protein. Repeat with a different name for the second protein.
3. Use the function once more with inputs of length 100, CATH_ANNOTATION of 3 (alpha beta), and a protein name for the first alpha beta protein. Repeat with a different name for the second protein.
4. Use the "analyze_protein_structure" function on each of the protein structures obtained.
5. Use the "calucalte_force_energy_from_seq" function to compute the maximum force and unfolding energy for each of the proteins.
6. Use the "calc_protein_ANM" function to compute the first 10 frequencies for each protein.

It can be seen that the planner demonstrates good performance in breaking the task into sub-tasks to be accomplished step by step. Moreover, it has identified and suggested the correct functions and corresponding input parameters for each sub-task. The plan is further supported by the “critic” who provides positive feedback as follows:

Critic (to chat_manager):

Yes, this plan sounds good. It covers each step required for the task. The parameter inputs for each function match with the required parameters as stated in their function definitions. Thus, this plan should successfully accomplish the task.

The multi-agent team then proceeds to execute the different steps outlined in the plan by calling and executing the functions. Specifically, the function 'design_protein_from_CATH' is executed with the appropriate 'CATH_ANNOTATION' for a specific protein structure design, as outlined in the plan. Following the generation of all proteins, the executions are followed by structural analysis and force and energy computations. It's noteworthy that the model exhibits good performance in restoring and memorizing the sequences of the generated proteins, which are essential for the force and energy calculations. Finally, the team successfully completes the task by computing the first 10 frequencies for each protein. An overview of the computations performed by the team of agents for this experiment is shown in Figure 4.

Given the complexity of the problem involving numerous computational tasks, a decent number of results have been generated in the first round of the conversation. In the next round, to evaluate the team's ability to memorize and restore the results, we present the following task:

user_proxy (to chat_manager):

Could you save the results in a CSV file named "protein_analysis.csv," where each row corresponds to a protein, and each column represents a specific property? Include the AA sequence in the results and use suitable identifiers for the columns.

In this task, we not only request the team to save the data but also require them to adhere to a customized format when storing the results. The model is proficient in creating a JSON dictionary that satisfies the specified format and saving the results to a CSV file, as illustrated in **Table 4**.

The plots of the obtained results are shown in Figure 4. The results indicate that Chroma has done a poor performance in creating β -rich protein named *mainly_beta_protein_2* which its structure is dominant in α -helix. As an attempt to test the capability of the multi-agent model in analyzing the results, in the last round of the conversation, we ask the model to assess Chroma's performance in generating the proteins conditioned on the secondary structure by posing the following question:

user_proxy (to chat_manager):

Based on the results of this example, can you check if the protein generator (Chroma) has been successful in creating proteins with desired structure?

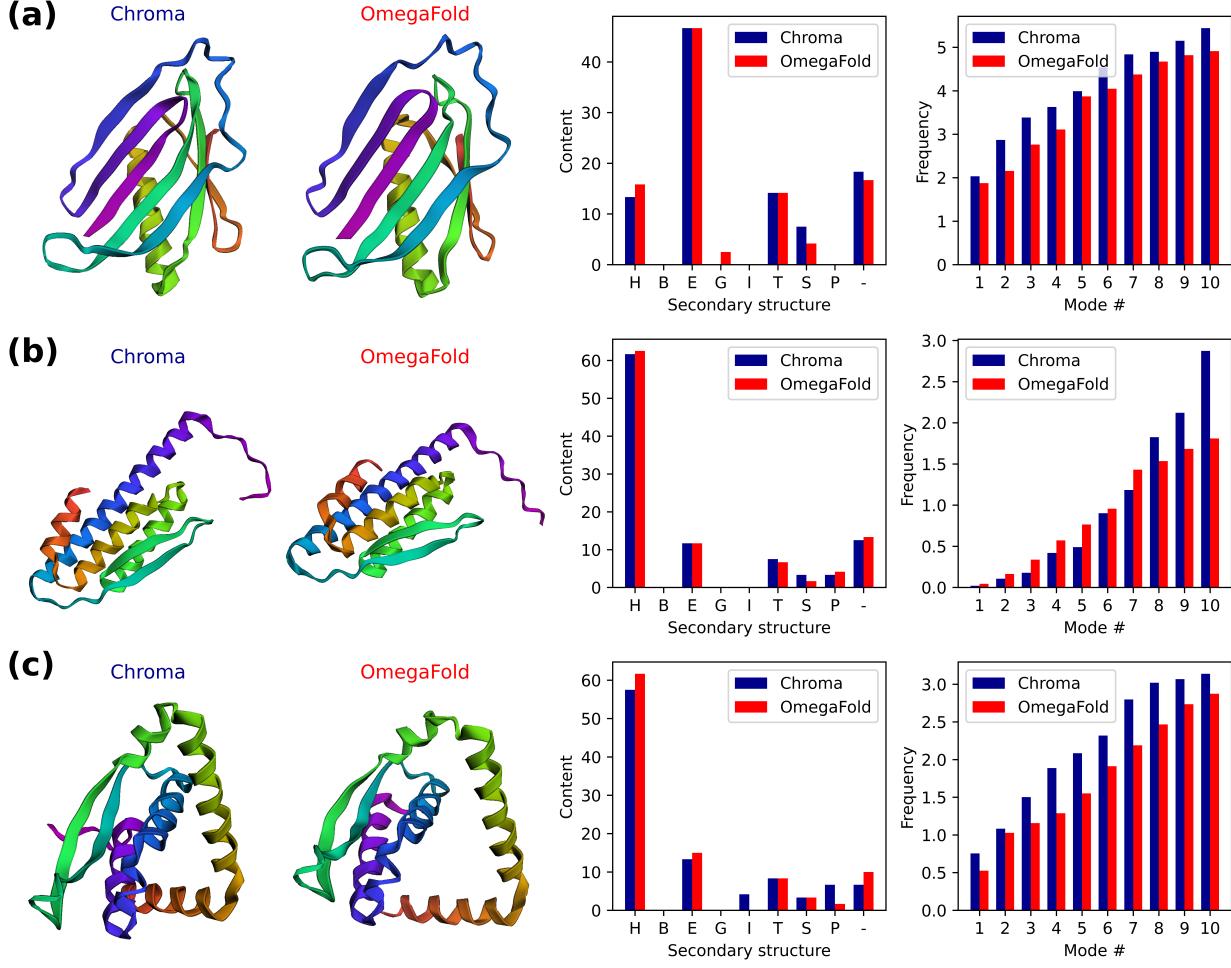


Figure 5: **The results generated by the multi-agent collaboration for the experiment II, Section 2.2.** The first and second columns depict the 3D folded structures of proteins generated by Chroma and OmegaFold2, respectively, while the third and fourth columns represent the fractional content of secondary structures, and first ten natural frequencies for the generated proteins.

The “critic” agent conducts a thorough evaluation of Chroma’s performance in generating proteins with targeted secondary structure content. Through a detailed analysis of each CATH structure, it reveals the inherent strengths and weaknesses in Chroma’s capabilities. Specifically, addressing the limitations of Chroma’s performance, the critic’s evaluation provides the following observations for the mainly beta proteins:

- The mainly beta proteins showed higher percentages of extended strand/beta-sheet secondary structure ('E'). Though, the percentages varied quite a bit (64% for mainly_beta_protein_1 and only 8% for mainly_beta_protein_2), which could be due to the complex nature of beta-structures.

This illustration not only highlights the multi-agent model’s proficiency in computational tasks but also underscores its intelligence in handling intricate data analyses—an aspect traditionally reserved for human. The full conversations for this experiment can be found in Table S4 of the Supplementary material.

3 Conclusions

Large Language Models (LLMs) have made remarkable strides, revealing their immense potential to potentially replicate human-like intelligence across diverse domains and modalities, demonstrating proficiency in comprehending extensive collective knowledge and proving adept at effectively applying this information. However, to reach intelligent

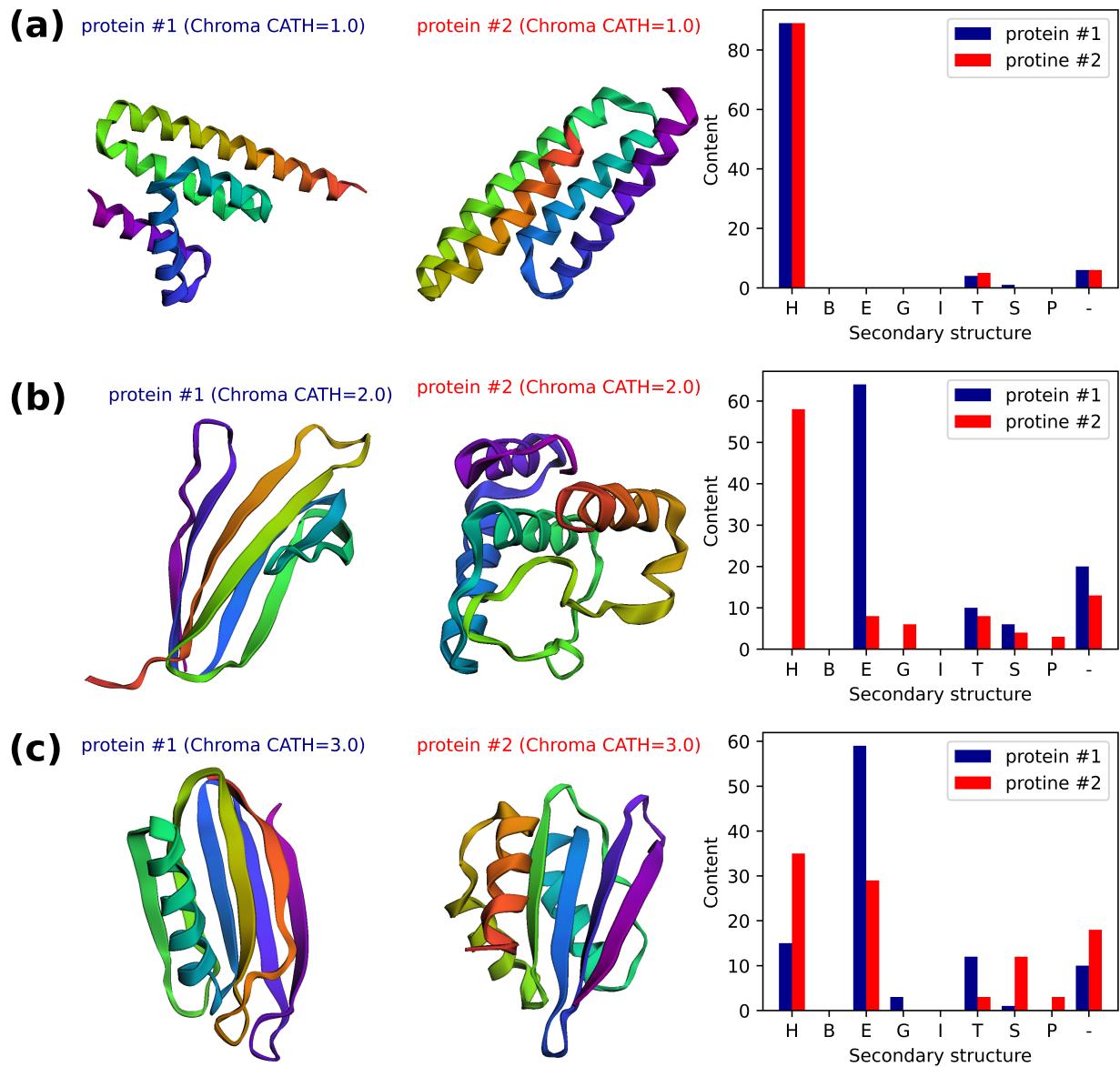


Figure 6: **The results generated by the multi-agent collaboration for the experiment III, Section 2.3.** The first and second columns depict the 3d folded structures and the last column represents the fractional content of secondary structures for the two proteins generated by Chroma conditioned on the CATH class of (a) 1: mainly alpha protein, (b) 2: mainly beta protein, and (c) 3: alpha beta protein.

Table 4: The final results generated by the group chat in the form of a CSV file, without human intervention, for the third experiment III, Section 2.3.

Protein Name #	AA Sequence	Secondary Structure	Unfolding Energy	Max Force	First 10 Frequencies
mainly_alpha_protein_1	SMKKIEDYIREKLKALGLSDEEI EERVKQLMEGIKNPKKFEKELQ KRNDRESLLIFKEAYALYEASK DKEKGKLINKVQSERDKWET EQAEAARAAAAA	'H': 89.0, 'B': 0.0, 'E': 0.0, 'G': 0.0, 'T': 0.0, 'I': 4.0, 'S': 1.0, 'P': 0.0, '-': 6.0	0.381	0.444	[0.2329, 0.4901, 0.9331, 1.3741, 1.734 7.2.1598, 2.3686, 2.6359, 2.8555, 3.03 64]
mainly_alpha_protein_2	MSKKIEELKKLLEDEVETLKEY ARQGDDACKKAADLIEEVKKA LEEGNPEKYSQLKKLTDANK AIEEYRKRFEEAGKPEAQKVID KLKKILDITIN	'H': 89.0, 'B': 0.0, 'E': 0.0, 'G': 0.0, 'I': 0.0, 'T': 5.0, 'S': 0.0, 'P': 0.0, '-': 6.0	0.376	0.536	[1.6126, 2.0783, 2.3073, 2.4565, 3.399, 3.475, 4.1377, 4.7104, 4.8864, 5.2187]
mainly_beta_protein_1	TTVTVPVADADGNEHSTVTA YGNKVTITITCPSCNTVTETVDG VAKTLGTSGNQTITETRTIAPD EVVTRYTCTPNASATSSKTQT VTIKGSQPAP	'H': 0.0, 'B': 0.0, 'E': 64.0, 'G': 0.0, 'I': 0.0, 'T': 10.0, 'S': 6.0, 'P': 0.0, '-': 20.0	0.462	0.533	[1.2806, 1.5057, 1.9846, 2.1025, 2.472 3.2.702, 2.9931, 3.1498, 3.4432, 4.168 5]
mainly_beta_protein_2	SLKAKNLEEMIKEAEKLGYRSR EVEKINNEIRDKFKLGVKISEKT LAYIAYLRLLGVKIDWDKIKKV KKATPADFRVSEEDLKPKIQKI LEKIKEIN	'H': 57.99, 'B': 0.0, 'E': 8.0, 'G': 6.0, 'I': 0.0, 'T': 8.0, 'S': 4.0, 'P': 3.0, '-': 13.0	0.371	0.548	[2.8864, 4.3752, 4.5928, 4.8295, 5.0854, 5.5618, 5.8646, 6.007, 6.3847, 7.1246]
alpha_beta_protein_1	APTVKTFEDTINGQKVTVTVA SPGGKITIKTSPGYGDEVAKAFI EELKKQNVLESYKVESAPGKET TISDVVKVSGATVTFYVINNGK KGKEYSVTVDA	'H': 15.0, 'B': 0.0, 'E': 59.0, 'G': 3.0, 'I': 0.0, 'T': 12.0, 'S': 1.0, 'P': 0.0, '-': 10.0	0.424	0.535	[2.4383, 2.5651, 3.3175, 3.8231, 3.967 3.4.2655, 4.6393, 5.1509, 5.6023, 5.95 55]
alpha_beta_protein_2	MELKVTEKKGKDGYKVKIEL NTPDKRYIHIIESDASRESLIKAAE ALLQGKEVEPTPVNEKNVVLFE DEDVKTTSIERSKKLFKSDNPEEN IKKALEYLLK	'H': 35.0, 'B': 0.0, 'E': 28.999999999999996, 'G': 0.0, 'I': 0.0, 'T': 3.0, 'S': 12.0, 'P': 3.0, '-': 18.0	0.376	0.543	[2.8756, 3.8895, 4.0594, 4.2831, 4.554 2.5.171, 5.3661, 5.4312, 6.1964, 6.306 6]

problem-solving systems, these types of models are not yet sufficient and require integration with other methods. In this study we explored the capability of AI agents to solve protein design problems in an autonomous manner without the need for human intervention. The agents have been powered by a general purpose LLM model, GPT-4, which allows them to communicate via conversation. It should be noted that the general capabilities of the AI agents powered by the LLM plays an important role at different stages of the problem solving. In our case, GPT-4-powered agents showed excellent proficiency specially in problem understanding, strategy development, and criticizing the outcomes. Such an AI system is not limited to mere linguistic interactions between agents; they have the capacity to incorporate a variety of special-purpose modeling and simulation tools, human input, tools for knowledge retrieval, and even deep learning-based surrogate models to solve particular tasks. Furthermore, additional tools can be integrated into the multi-agent system with popular external APIs and up-to-date knowledge about special topics can be retrieved by searching and browsing the web through specialized API interfaces. By harnessing the collective abilities of agents, including reasoning, tool usage, criticism, mutual correction, adaptation to new observations, and communication this framework has proven highly effective in navigating intricate challenges including protein design.

To achieve this goal we constructed a group of agents, each assigned a unique profile through initial prompts, to dynamically interact in a group chat via conversations and make decisions and take actions based on their observations. The agents profile outlines their attributes, roles, and functionalities within the system and describe communication protocols to exchange information with other agents in the system. Our team of agents include a user proxy to pose the query, a planner to formulate a plan, an function-baked assistant to execute the functions, and a critic to evaluate the outcome and criticizing the performance. We also use a chat manager to lead the group chat by dynamically choosing the working agent based on the current outcome and the agents' roles. Through a series of experiments, we unleashed the power of agents in not only conducting the roles they were assigned to, but to autonomously collaborate by discussion powered by the all-purpose LLM. For example, the agent playing the role of a planner successfully identified all the tasks in the query and suggested a details plan including the necessary functions to accomplish them. Furthermore, the agent assigned the critic role, is able to give constructive feedback about the plan or provide suggestions in case of failure, to correct errors that may emerge. Our experiments have showcased the great potential of the multi-agent modeling framework in tackling complex tasks as well as integrating AI-agents into physics-based modeling.

Multi-agent modeling is a powerful technique that offers enhanced problem-solving capacity as shown here in various computational experiments in the realm of protein design, physics modeling, and analysis. Given a complex query comprising multi-objective tasks, using the idea of division of labor, the model excels at developing a strategy to break the task into sub-tasks and then, recruiting a set of agents to effectively engage in problem solving tasks in an

autonomous fashion. Tool-backed agents have the capacity to execute tools via function execution. We equipped an agent with a rich library of tools that span a broad spectrum of functionalities including *de novo* protein design, protein folding, and protein secondary structure analysis among others. The fact that there is no intrinsic limitation in customizing the functions, allows us to integrate knowledge across different disciplines into our model and analysis, for instance by integrating knowledge retrieval systems or retrieving physical data via simulations. For instance, here we utilized coarse grained simulations to obtain natural frequencies of proteins but the model offers a high flexibility in defining functions that focus on other particular area simulation (e.g. an expert in performing Density Functional Theory, Molecular Dynamics, or even physics-inspired neural network solvers[71, 72, 37]). Multi-agent framework can also accelerate the discovery of *de novo* proteins with targeted mechanical properties by embracing the power of robust end-to-end deep models solving forward and inverse protein design problems[24, 59, 17, 73, 74, 75, 76]

Developing these models that connect some structural protein features, such as secondary structure, to a material property, such as toughness or strength have gained a lot of attention recently. Here, we used a pre-trained autoregressive transformer model to predict the maximum force and energy of protein unfolding, but other end-to-end models could also be utilized. In the context of inverse protein design problems, a team of two agents, one expert in the forward tasks and the other in the inverse task, can be collaborated to assist the cycle check wherein the *de novo* proteins certainly meet the specified property criteria. Along the same line, one could benefit from the multi-agent collaboration in evaluating the accuracy of generative models in conditional designing of proteins or compare the created 3D structures with the state-of-the-art folding tools[16, 77, 78]. For example, through an automated process of protein generation and structure analysis, our ProtAgents framework revealed the shortcomings of Chroma in designing β -sheet-rich proteins. In another example, the folded 3D structures of Chroma were compared with those obtained by OmegaFold2. All these examples, demonstrate the capacity of multi-agent framework in a wide range of applications in the context of protein design and analysis. Lastly, the model enables integrating various information across scales, whether new protein sequences or physics simulations output in form of rich data structures, for inclusion in easily readable file formats (like JSON) to be used by other agents or to be stored for future analysis.

Designing *de novo* proteins that meet special objectives in term of mechanical or structural properties present unique challenges calling for new strategies. The prevailing strategies often rely on developing data-driven end-to-end deep learning models to find the complex mapping from protein constitutive structure to property or vice versa. However, these models often focus on specific properties, limiting their functionality in multi-objective design purposes where several criteria needs to be met. To overcome these challenges and propel the field forward, future research endeavors could revolve around the development of an integrated system of agents designed to automate the entire lifecycle of training deep neural networks for protein design. Each agent within this system could be assigned specific responsibilities, such as data generation through simulations, data curation for ensuring quality and relevance, and the execution of the code required for model training. Additionally, a critic agent could monitor and critique the training process, making decisions like early stopping or tuning hyperparameters to enhance the model's accuracy. This collaborative and automated approach would not only streamline the design process but also contribute to achieving higher or desired levels of accuracy in the generated models. Furthermore, this agent-based strategy can extend to on-the-fly active learning, where agents dynamically adapt the model based on real-time feedback, improving its performance iteratively. By incorporating intelligent agents at every stage of the process, the proposed system aims to revolutionize the landscape of *de novo* protein design, making it more efficient, adaptive, and capable of meeting diverse and complex design objectives, offering a new paradigm in materials design workflows.

4 Materials and Methods

Agent design

As shown in Figure 1a, we design AI agents using the state-of-the-art all-purpose LLM GPT-4 and dynamic multi-agent collaboration is implemented in AutoGen framework[79], an open-source ecosystem for agent-based AI modeling. Additional agents are introduced as described below, including some based on generative AI as well as physics modeling.

In our multi-agent system, the human *user_proxy* agent is constructed using UserProxyAgent class from Autogen, and *Assistant*, *Planner*, *Critic* agents are created via AssistantAgent class from Autogen; and the group chat manager is created using GroupChatManager class. Each agent is assigned a role through a profile description listed in **Table 1**, included as *system_message* at their creation.

Function and tool design

All the tools implemented in this work are defined as python functions. Each function is characterized by a name, a description, and input properties with a description as tabulated in **Table S1** of the Supplementary Material. The list of

functions are incorporated into the multi-agent system, included as the *function_map* parameter in the Assistant agent at its creation.

Autoregressive transformer model to predict protein unfolding force-extension from sequences

We use a special-purpose GPT-style model denoted as ProteinForceGPT, similar as in , here trained to predict force-extension curves from sequences along with other mechanical properties, and vice versa (<https://huggingface.co/lamm-mit/ProteinForceGPT>). The protein language model is based on the NeoGPT-X architecture and uses rotary positional embeddings (RoPE)[80]. The model has 16 attention heads, 36 hidden layers and a hidden size of 1024, an intermediate size of 4096 and uses GeLU activation functions.

Pre-training was conducted based on a dataset of $\sim 800,000$ amino acid sequences, using next-token predictions using a “Sequence” task (<https://huggingface.co/datasets/lamm-mit/GPTProteinPretrained>):

```
Sequence<GEECDCGSPSNPCCDAATCKLRPGAQCADGLCCDQCRCFKKRTICRIARGDFPDDRCTGQSADCPRN>
```

The ProteinForceGPT model was then fine-tuned bidirectionally, to predict mechanical properties of proteins from their sequence, as well as sequence candidates that meet a required force-extension behavior and various other properties. Fine-tuning is conducted using a dataset derived from molecular dynamics (MD) simulations[81]. Sample tasks for the model include:

```
CalculateForce<GEECDCGSPSNPCCDAATCKLRPGAQCADGLCCDQCRCFKKRTICRIARGDFPDDRCTGQSADCPRN> [0.262]
CalculateEnergy<GEECDCGSPSNPCCDAATCKLRPGAQCADGLCCDQCRCFKKRTICRIARGDFPDDRCTGQSADCPRN> [0.220]
CalculateForceEnergy<GEECDCGSPSNPCCDAATCKLRPGAQCADGLCCDQCRCFKKRTICRIARGDFPDDRCTGQSADCPRN> [0.262,0.220]
CalculateForceHistory<GEECDCGSPSNPCCDAATCKLRPGAQCADGLCCDQCRCFKKRTICRIARGDFPDDRCTGQSADCPRN>
[0.004,0.034,0.125,0.142,0.159,0.102,0.079,0.073, 0.131,0.105,0.071,0.058,0.072,0.060,0.049,0.114,
0.122,0.108,0.173,0.192,0.208,0.153,0.212,0.222, 0.244]
GenerateForce<0.262> [GEECDCGSPSNPCCDAATCKLRPGAQCADGLCCDQCRCFKKRTICRIARGDFPDDRCTGQSADCPRN]
GenerateForce<0.220> [GEECDCGSPSNPCCDAATCKL RPGAQCADGLCCDQCRCFKKRTICRIARGDFPDDRCTGQSADCPRN]
GenerateForceEnergy<0.262,0.220> [GEECDCGSPSNPCCDAATCKLRPGAQCADGLCCDQCRCFKKRTICRIARGDFPDDRCTGQSADCPRN]
GenerateForceHistory<0.004,0.034,0.125,0.142,0.159, 0.102,0.079,0.073,0.131,0.105,0.071,0.058,0.072,
0.060, 0.049,0.114,0.122,0.108,0.173,0.192,0.208, 0.153,0.212, 0.222,0.244> [GEECDCGSPSNPCCDAATCKLR
PQAQCADGLCCDQCRCFKKRTICRIARGDFPDDRCTGQSADCPRN]
```

Sample results from validation of the model are shown in **Figure S2**. We only use forward predictions for use in the agent model reported here.

Software versions and hardware

We develop our multi-agent models using local workstations with NVIDIA GPUs. We use Python 3.10 and pyautogen-0.2.2[82]. Additional implementation details are included in the code.

Visualization

We use Py3DMol[83] for visualization of the protein structures.

Secondary Structure Analysis

We use the dictionary of protein secondary structure (DSSP)[84] module via BioPython[85] to analyze the secondary structure content of the proteins from its geometry.

Natural Vibrational Frequency Calculations

We perform Anisotropic Network Model (ANM)[86, 87] calculations as implemented in ProDy[88] for normal mode analysis. The problem is solved by considering the protein as a network of interactions, defined within a cutoff distance for which spring-like potentials are assumed to define molecular interactions.

Retrieval Augmented Generation

We use Llama Index[89] as a tool to implement RAG where the full text of papers cited as references [65, 66] are used as external sources from which information can be retrieved by the system in real-time.

Conflict of interest

The author declares no conflict of interest.

Data and code availability

All data and codes are available on GitHub at <https://github.com/lamm-mit/ProtAgents>. Alternatively, they will be provided by the corresponding author based on reasonable request.

Author Contributions: MJB and AG conceived the study and developed the multi-agent models. AG performed the tests for various problems, analyzed the results and prepared the first draft of the paper. MJB supported the analysis, revised and finalized the paper with AG.

Supplementary Materials

The full records of different conversation experiments along with additional materials are provided as Supplementary Materials.

Acknowledgements

We acknowledge support from USDA (2021-69012-35978), DOE-SERDP (WP22-S1-3475), ARO (79058LSCSB, W911NF-22-2-0213 and W911NF2120130) as well as the MIT-IBM Watson AI Lab, MIT's Generative AI Initiative, and Google. Additional support from NIH (U01EB014976 and R01AR077793) ONR (N00014-19-1-2375 and N00014-20-1-2189) is acknowledged. AG gratefully acknowledges the financial support from the Swiss National Science Foundation (#P500PT_214448).

References

- [1] Po Ssu Huang, Scott E. Boyken, and David Baker. The coming of age of de novo protein design. *Nature* 2016 537:7620, 537:320–327, 9 2016.
- [2] Pascal Notin, Mafalda Dias, Jonathan Frazer, Javier Marchena Hurtado, Aidan N Gomez, Debora Marks, and Yarin Gal. Tranception: Protein fitness prediction with autoregressive transformers and inference-time retrieval, 6 2022.
- [3] John Ingraham, Vikas K Garg, Regina Barzilay, and Tommi Jaakkola. Generative models for graph-based protein design. *Advances in Neural Information Processing Systems*, 32, 2019.
- [4] Kevin E. Wu, Kevin K. Yang, Rianne van den Berg, James Y. Zou, Alex X. Lu, and Ava P. Amini. Protein structure generation via folding diffusion. <https://arxiv.org/abs/2209.15611v2>, 9 2022.
- [5] Namrata Anand and Tudor Achim. Protein structure and sequence generation with equivariant denoising diffusion probabilistic models. <https://arxiv.org/abs/2205.15019v1>, 5 2022.
- [6] Raphael R. Eguchi, Christian A. Choe, and Po Ssu Huang. Ig-vae: Generative modeling of protein structure by direct 3d coordinate generation. *PLOS Computational Biology*, 18:e1010271, 6 2022.
- [7] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences of the United States of America*, 118:e2016239118, 4 2021.
- [8] Ali Madani, Bryan McCann, Nikhil Naik, Nitish Shirish Keskar, Namrata Anand, Raphael R. Eguchi, Po-Ssu Huang, and Richard Socher. Progen: Language modeling for protein generation. <https://arxiv.org/abs/2004.03497v1>, 3 2020.
- [9] Namrata Anand, Raphael Eguchi, Irimpan I. Mathews, Carla P. Perez, Alexander Derry, Russ B. Altman, and Po Ssu Huang. Protein sequence design with a learned potential. *Nature Communications* 2022 13:1, 13:1–11, 2 2022.
- [10] Joe G. Greener, Lewis Moffat, and David T. Jones. Design of metalloproteins and novel protein folds using variational autoencoders. *Scientific Reports* 2018 8:1, 8:1–12, 11 2018.
- [11] Adam J. Riesselman, John B. Ingraham, and Debora S. Marks. Deep generative models of genetic variation capture the effects of mutations. *Nature Methods* 2018 15:10, 15:816–822, 9 2018.

- [12] Ethan C. Alley, Grigory Khimulya, Surojit Biswas, Mohammed AlQuraishi, and George M. Church. Unified rational protein engineering with sequence-based deep representation learning. *Nature Methods* 2019 16:12, 16:1315–1322, 10 2019.
- [13] Joseph L. Watson, David Juergens, Nathaniel R. Bennett, Brian L. Trippe, Jason Yim, Helen E. Eisenach, Woody Ahern, Andrew J. Borst, Robert J. Ragotte, Lukas F. Milles, Basile I.M. Wicky, Nikita Hanikel, Samuel J. Pellock, Alexis Courbet, William Sheffler, Jue Wang, Preetham Venkatesh, Isaac Sappington, Susana Vázquez Torres, Anna Lauko, Valentin De Bortoli, Emile Mathieu, Sergey Ovchinnikov, Regina Barzilay, Tommi S. Jaakkola, Frank DiMaio, Minkyung Baek, and David Baker. De novo design of protein structure and function with rfdiffusion. *Nature* 2023 620:7976, 620:1089–1100, 7 2023.
- [14] Ivan Anishchenko, Samuel J. Pellock, Tamuka M. Chidyausiku, Theresa A. Ramelot, Sergey Ovchinnikov, Jingzhou Hao, Khushboo Bafna, Christoffer Norn, Alex Kang, Asim K. Bera, Frank DiMaio, Lauren Carter, Cameron M. Chow, Gaetano T. Montelione, and David Baker. De novo protein design by deep network hallucination. *Nature* 2021 600:7889, 600:547–552, 12 2021.
- [15] John B. Ingraham, Max Baranov, Zak Costello, Karl W. Barber, Wujie Wang, Ahmed Ismail, Vincent Frappier, Dana M. Lord, Christopher Ng-Thow-Hing, Erik R. Van Vlack, Shan Tie, Vincent Xue, Sarah C. Cowles, Alan Leung, João V. Rodrigues, Claudio L. Morales-Perez, Alex M. Ayoub, Robin Green, Katherine Puentes, Frank Oplinger, Nishant V. Panwar, Fritz Obermeyer, Adam R. Root, Andrew L. Beam, Frank J. Poelwijk, and Gevorg Grigoryan. Illuminating protein space with a programmable generative model. *Nature* 2023 623:7989, 623:1070–1078, 11 2023.
- [16] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A.A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishabh Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature* 2021 596:7873, 596:583–589, 7 2021.
- [17] Chi Hua Yu, Wei Chen, Yu Hsuan Chiang, Kai Guo, Zaira Martin Moldes, David L. Kaplan, and Markus J. Buehler. End-to-end deep learning model to predict and design secondary structure content of structural proteins. *ACS Biomaterials Science and Engineering*, 8:1156–1165, 3 2022.
- [18] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, Debsindhu Bhowmik, and Burkhard Rost. Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:7112–7127, 10 2022.
- [19] Claudio Mirabello and Gianluca Pollastri. Porter, paleale 4.0: high-accuracy prediction of protein secondary structure and relative solvent accessibility. *Bioinformatics*, 29:2056–2058, 8 2013.
- [20] Gianluca Pollastri, Darisz Przybylski, Burkhard Rost, and Pierre Baldi. Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles. *Proteins: Structure, Function, and Bioinformatics*, 47:228–235, 5 2002.
- [21] Buzhong Zhang, Jinyan Li, and Qiang Lü. Prediction of 8-state protein secondary structures by a novel deep learning architecture. *BMC Bioinformatics*, 19:1–13, 8 2018.
- [22] Gianluca Pollastri and Aoife McLysaght. Porter: a new, accurate server for protein secondary structure prediction. *Bioinformatics*, 21:1719–1720, 4 2005.
- [23] Kai Guo and Markus J. Buehler. Rapid prediction of protein natural frequencies using graph neural networks. *Digital Discovery*, 1:277–285, 6 2022.
- [24] Yiwen Hu and Markus J. Buehler. End-to-end protein normal mode frequency predictions using language and graph models and application to sonification. *ACS Nano*, 16:20656–20670, 12 2022.
- [25] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. 6 2022.
- [26] Yupeng Chang, X U Wang, Yuan Wu, Hao Chen, Wei Ye, Yue Zhang, Y I Chang, Philip S Yu, Xiaoyuan Yi, Yi Chang, Qiang Yang, Hong Kong, Xu Wang, Jindong Wang, Linyi Yang, Kaijie Zhu, Cunxiang Wang, Yidong Wang, and Xing Xie. A survey on evaluation of large language models. *J. ACM*, 37:42, 7 2023.
- [27] Kevin Maik Jablonka, Qianxiang Ai, Alexander Al-Feghali, Shruti Badhwar, Joshua D. Bocarsly, Andres M. Bran, Stefan Bringuier, L. Catherine Brinson, Kamal Choudhary, Defne Circi, Sam Cox, Wibe A. de Jong, Matthew L.

- Evans, Nicolas Gastellu, Jerome Genzling, María Victoria Gil, Ankur K. Gupta, Zhi Hong, Alishba Imran, Sabine Kruschwitz, Anne Labarre, Jakub Lála, Tao Liu, Steven Ma, Sauradeep Majumdar, Garrett W. Merz, Nicolas Moitessier, Elias Moubarak, Beatriz Mouriño, Brenden Pelkie, Michael Pieler, Mayk Caldas Ramos, Bojana Ranković, Samuel G. Rodrigues, Jacob N. Sanders, Philippe Schwaller, Marcus Schwarting, Jiale Shi, Berend Smit, Ben E. Smith, Joren Van Herck, Christoph Völker, Logan Ward, Sean Warren, Benjamin Weiser, Sylvester Zhang, Xiaoqi Zhang, Ghezal Ahmad Zia, Aristana Scourtas, K. J. Schmidt, Ian Foster, Andrew D. White, and Ben Blaiszik. 14 examples of how LLMs can transform materials science and chemistry: a reflection on a large language model hackathon. *Digital Discovery*, 2:1233–1250, 10 2023.
- [28] Markus J. Buehler. Melm, a generative pretrained language modeling framework that solves forward and inverse mechanics problems. *Journal of the Mechanics and Physics of Solids*, 181:105454, 12 2023.
 - [29] Markus J. Buehler. Mechgpt, a language-based strategy for mechanics and materials modeling that connects knowledge across scales, disciplines and modalities. *Applied Mechanics Reviews*, pages 1–82, 10 2023.
 - [30] James Boyko, Joseph Cohen, Nathan Fox, Maria Han Veiga, Jennifer I-Hsiu Li, Jing Liu, Bernardo Modenesi, Andreas H. Rauch, Kenneth N. Reid, Soumi Tribedi, Anastasia Visheratina, and Xin Xie. An interdisciplinary outlook on large language models for scientific research. 11 2023.
 - [31] Bo Ni and Markus J. Buehler. Mechagents: Large language model multi-agent collaborations can solve mechanics problems, generate new data, and integrate knowledge. 2023.
 - [32] Neal R. Brodnik, Samuel Carton, Caelin Muir, Satanu Ghosh, Doug Downey, McLean P. Echlin, Tresa M. Pollock, and Samantha Daly. Perspective: Large language models in applied mechanics. *Journal of Applied Mechanics*, 90, 10 2023.
 - [33] Robert Tinn, Hao Cheng, Yu Gu, Tristan Naumann, Jianfeng Gao, Hoifung Poon Correspondence, Naoto Usuyama, Xiaodong Liu, and Hoifung Poon. Fine-tuning large neural language models for biomedical natural language processing highlights d systematic exploration of fine-tuning stability in biomedical nlp d domain-specific vocabulary and pretraining facilitate robust models for fine-tuning d pubmedbert-large and pubmedelectra models advance state-of-the-art in biomedical nlp in brief fine-tuning large neural language models for biomedical natural language processing. *Patterns*, 4:100729, 2023.
 - [34] Yiwen Hu and Markus J. Buehler. Deep language models for interpretative and predictive materials science. *APL Machine Learning*, 1:10901, 3 2023.
 - [35] Markus J. Buehler. Generative retrieval-augmented ontologic graph and multi-agent strategies for interpretive large language model-based materials design. *ACS Engineering AU*, 10 2023.
 - [36] Rachel K. Luu and Markus J. Buehler. Materials informatics tools in the context of bio-inspired material mechanics. *Journal of Applied Mechanics*, 90, 9 2023.
 - [37] Grace C.Y. Peng, Mark Alber, Adrian Buganza Tepole, William R. Cannon, Suvarnu De, Salvador Dura-Bernal, Krishna Garikipati, George Karniadakis, William W. Lytton, Paris Perdikaris, Linda Petzold, and Ellen Kuhl. Multiscale modeling meets machine learning: What can we learn? *Archives of Computational Methods in Engineering*, 28:1017–1037, 5 2021.
 - [38] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.
 - [39] Madeleine Bates. Models of natural language understanding. *Proceedings of the National Academy of Sciences*, 92:9977–9982, 10 1995.
 - [40] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Vincent Zhao, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Pranesh Srinivasan, Laichee Man, Kathleen Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed Chi, and Quoc Le. Lamda: Language models for dialog applications. 1 2022.

- [41] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24:1–113, 2023.
- [42] Ashish Vaswani, Google Brain, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.
- [43] OpenAI and: Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O’Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emry Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report. 3 2023.
- [44] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave

- Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. 7 2021.
- [45] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of Artificial General Intelligence: Early experiments with GPT-4. 3 2023.
 - [46] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.
 - [47] Kai Guo, Zhenze Yang, Chi Hua Yu, and Markus J. Buehler. Artificial intelligence and machine learning in design of mechanical materials. *Materials Horizons*, 8:1153–1172, 4 2021.
 - [48] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33:6999–7019, 12 2022.
 - [49] Keiron O’Shea and Ryan Nash. An introduction to convolutional neural networks. *International Journal for Research in Applied Science and Engineering Technology*, 10:943–947, 11 2015.
 - [50] Eric L. Buehler and Markus J. Buehler. End-to-end prediction of multimaterial stress fields and fracture patterns using cycle-consistent adversarial and transformer neural networks. *Biomedical Engineering Advances*, 4:100038, 12 2022.
 - [51] Zhenze Yang, Chi Hua Yu, and Markus J. Buehler. Deep learning model to predict complex stress and strain fields in hierarchical composites. *Science Advances*, 7, 4 2021.
 - [52] Donatas Repecka, Vykintas Jauniskis, Laurynas Karpus, Elzbieta Rembeza, Irmantas Rokaitis, Jan Zrimec, Simona Poviloniene, Audrius Laurynenas, Sandra Viknander, Wissam Abuajwa, Otto Savolainen, Rolandas Meskys, Martin K.M. Engqvist, and Aleksej Zelezniak. Expanding functional protein sequence spaces using generative adversarial networks. *Nature Machine Intelligence* 2021 3:4, 3:324–333, 3 2021.
 - [53] Renzhi Cao, Colton Freitas, Leong Chan, Miao Sun, Haiqing Jiang, and Zhangxin Chen. Prolango: Protein function prediction using neural machine translation based on a recurrent neural network. *Molecules* 2017, Vol. 22, Page 1732, 22:1732, 10 2017.
 - [54] Yu-Chuan Hsu, Chi-Hua Yu, and Markus J Buehler. Using deep learning to predict fracture patterns in crystalline solids. *Matter*, 3:197–211, 2020.
 - [55] Wei Lu, Zhenze Yang, and Markus J. Buehler. Rapid mechanical property prediction and de novo design of three-dimensional spider webs through graph and graphperceiver neural networks. *Journal of Applied Physics*, 132:74703, 8 2022.
 - [56] Alexey Strokach, David Becerra, Carles Corbi-Verge, Albert Perez-Riba, and Philip M Kim. Fast and flexible protein design using deep graph neural networks. *Cell Systems*, 11:402–411.e4, 2020.
 - [57] Ronghui You, Shuwei Yao, Hiroshi Mamitsuka, and Shanfeng Zhu. Deepgraphgo: graph neural network for large-scale, multispecies protein function prediction. *Bioinformatics*, 37:i262–i271, 7 2021.
 - [58] Zhenze Yang and Markus J. Buehler. Linking atomic structural defects to mesoscale properties in crystalline solids using graph neural networks. *npj Computational Materials* 2022 8:1, 8:1–13, 9 2022.
 - [59] Wei Lu, David L. Kaplan, and Markus J. Buehler. Generative modeling, design, and analysis of spider silk protein sequences for enhanced mechanical properties. *Advanced Functional Materials*, page 2311324, 12 2023.
 - [60] Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B. Tenenbaum, Tianmin Shu, and Chuang Gan. Building cooperative embodied agents modularly with large language models. 7 2023.
 - [61] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. A survey on large language model based autonomous agents. 8 2023.
 - [62] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, and Tao Gui. The rise and potential of large language model based agents: A survey. 9 2023.

- [63] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.
- [64] Openai api.
- [65] Mateusz Sikora, Joanna I. Sułkowska, and Marek Cieplak. Mechanical strength of 17 134 model proteins and cysteine slipknots. *PLOS Computational Biology*, 5:e1000547, 2009.
- [66] Arata Nakajo, Zacharie Wuillemin, Patrick Metzger, al, Yuxiu Liu, Michael W Murphy, Daniel R Baker, Ana Marijja Damjanovi, Burak Koyutürk, Yan-Sheng Li, Joanna I Sułkowska, and Marek Cieplak. Mechanical stretching of proteins—a theoretical survey of the protein data bank. *Journal of Physics: Condensed Matter*, 19:283201, 6 2007.
- [67] Theodor Ackbarow, Xuefeng Chen, Sinan Keten, and Markus J. Buehler. Hierarchies, multiple energy barriers, and robustness govern the fracture mechanics of α -helical and β -sheet protein domains. *Proceedings of the National Academy of Sciences of the United States of America*, 104:16410–16415, 10 2007.
- [68] Tuomas P.J. Knowles and Markus J. Buehler. Nanomechanics of functional and pathological amyloid materials. *Nature Nanotechnology* 2011 6:8, 6:469–479, 7 2011.
- [69] Zhiping Xu and Markus J. Buehler. Mechanical energy transfer and dissipation in fibrous beta-sheet-rich proteins. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 81:061910, 6 2010.
- [70] Zhao Qin and Markus J. Buehler. Cooperative deformation of hydrogen bonds in beta-strands and beta-sheet nanocrystals. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 82:061906, 12 2010.
- [71] Isaac Elias Lagaris, Aristidis Likas, and Dimitrios I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9:987–1000, 1998.
- [72] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2 2019.
- [73] Frank Y.C. Liu, Bo Ni, and Markus J. Buehler. Presto: Rapid protein mechanical strength prediction with an end-to-end deep learning model. *Extreme Mechanics Letters*, 55:101803, 8 2022.
- [74] Chi Hua Yu and Markus J. Buehler. Sonification based de novo protein design using artificial intelligence, structure prediction, and analysis using molecular modeling. *APL Bioengineering*, 4:16108, 3 2020.
- [75] Markus J. Buehler. Generative pretrained autoregressive transformer graph neural network applied to the analysis and discovery of novel proteins. *Journal of Applied Physics*, 134, 5 2023.
- [76] Bo Ni, David L Kaplan, and Markus J Buehler. Generative design of de novo proteins based on secondary-structure constraints using an attention-based diffusion model. *Chem*, 9:1828–1849, 2023.
- [77] Mohammed AlQuraishi. Machine learning in protein structure prediction. *Current Opinion in Chemical Biology*, 65:1–8, 12 2021.
- [78] Wenhao Gao, Sai Pooja Mahajan, Jeremias Sulam, and Jeffrey J Gray. Deep learning in protein structural modeling and design. *Patterns*, 1:100142, 2020.
- [79] Chenxu Zhu, Bo Chen, Huifeng Guo, Hang Xu, Xiangyang Li, Xiangyu Zhao, Weinan Zhang, Yong Yu, and Ruiming Tang. Autogen: An automated dynamic model generation framework for recommender system. *WSDM 2023 - Proceedings of the 16th ACM International Conference on Web Search and Data Mining*, pages 598–606, 2 2023.
- [80] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.
- [81] Bo Ni, David L. Kaplan, and Markus J. Buehler. Forcegen: End-to-end de novo protein generation based on nonlinear mechanical unfolding responses using a protein language diffusion model. *Science Advances*, 2023.
- [82] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Awadallah, Ryen W White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen LLM applications via multi-agent conversation. <https://arxiv.org/abs/2308.08155>, 2023.
- [83] Nicholas Rego and David Koes. 3dmol.js: molecular visualization with webgl. *Bioinformatics*, 31:1322–1324, 4 2015.
- [84] Wolfgang Kabsch and Christian Sander. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12):2577–2637, 12 1983.

- [85] Peter J.A. Cock, Tiago Antao, Jeffrey T. Chang, Brad A. Chapman, Cymon J. Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, and Michiel J.L. De Hoon. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25:1422, 6 2009.
- [86] A R Atilgan, S R Durell, R L Jernigan, M C Demirel, O Keskin, and I Bahar. Anisotropy of fluctuation dynamics of proteins with an elastic network model. *Biophys. J.*, 2001.
- [87] Pemra Doruker, Ali Rana Atilgan, and Ivet Bahar. Dynamics of proteins predicted by molecular dynamics simulations and analytical approaches: Application to-amylase inhibitor. *PROTEINS: Structure, Function, and Bioinformatics*, 2000.
- [88] Ahmet Bakan, Lidio M. Meireles, and Ivet Bahar. Prody: Protein dynamics inferred from theory and experiments. *Bioinformatics*, 27:1575–1577, 6 2011.
- [89] LlamaIndex (formerly GPT index), a data framework for LLM applications.